A Semantic Framework for Test Coverage ⋆

Laura Brandán Briones[+], Ed Brinksma[+*], and Mariëlle Stoelinga[+]

[+] Faculty of Computer Science, University of Twente, The Netherlands
[*] Embedded Systems Institute, The Netherlands
{marielle,brandanl}@cs.utwente.nl, Ed.Brinksma@esi.nl

Abstract. Since testing is inherently incomplete, test selection has vital importance. Coverage measures evaluate the quality of a test suite and help the tester select test cases with maximal impact at minimum cost. Existing coverage criteria for test suites are usually defined in terms of syntactic characteristics of the implementation under test or its specification. Typical black-box coverage metrics are state and transition coverage of the specification. White-box testing often considers statement, condition and path coverage. A disadvantage of this syntactic approach is that different coverage figures are assigned to systems that are behaviorally equivalent, but syntactically different. Moreover, those coverage metrics do not take into account that certain failures are more severe than others, and that more testing effort should be devoted to uncover the most important bugs, while less critical system parts can be tested less thoroughly.
This paper introduces a semantic approach to black box test coverage. Our starting point is a weighted fault model (or WFM), which augments a specification by assigning a weight to each error that may occur in an implementation. We define a framework to express coverage measures that express how well a test suite covers such a specification, taking into account the error weight. Since our notions are semantic, they are insensitive to replacing a specification by one with equivalent behaviour. We present several algorithms that, given a certain minimality criterion, compute a minimal test suite with maximal coverage. These algorithms work on a syntactic representation of WFMs as fault automata. They are based on existing and novel optimization problems. Finally, we illustrate our approach by analyzing and comparing a number of test suites for a chat protocol.

1 Introduction

After years of limited attention, the theory of testing has now become a widely studied, academically respectable subject of research. In particular, the application of formal methods in the area of model-driven testing has led to a better understanding of the notion of conformance between an implementation and a specification. Automated generation methods for test suites from specifications

⋆ This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.000.505 (MOQS); the EU under grant number IST-004527 (ARTIST2); and by the DFG/NWO bilateral cooperation programme under project number DN 62-600 (VOSS2).

[15, 16, 4, 13] have been developed, which have lead to a new generation of powerful test generation and execution tools such as SpecExplorer[6], TorX[3] and TGV[8].

A clear advantage of a formal approach to testing is the provable soundness of the generated test suites, i.e. the property that each generated test suite will only reject implementations that do not conform to the given specification. In many cases also a completeness or exhaustiveness result is obtained, i.e. the property that for each non-conforming implementation a test case can be generated that will expose its errors by rejecting it (cf. [15]).

In practice, the above notion of exhaustiveness is usually problematic, since exhaustive test suites will contain infinitely many tests. This raises the question of test selection, i.e. the selection of well-chosen, finite test suites that can be generated (and executed) within the available resources. Test case selection is naturally related to a measure of coverage, indicating how much of the required conformance is tested for by a given test selection. In this way, coverage measures can assist the tester in choosing test cases with maximal impact against some optimization criterion (i.e. number of tests, execution time, cost).

Typical coverage measures used in black-box testing are the number of states and/or transitions of the specification that would be visited by executing a test suite against it [17, 9, 12]; white-box testing often considers the number of statements, conditional branches, and paths through the implementation code that are touched by the test suite execution [10, 11, 1]. Although these measures do indeed help with the selection of tests and the exposure of faults, they share two shortcomings:

1. The approaches are based on syntactic model features, i.e. coverage figures are based on constructs of the specific model or program used as a reference. Therefore, we may get different coverage results when we replace the model in question with a behaviorally equivalent, but syntactically different one.
2. The approaches fail to account for the non-uniform gravity of failures, whereas it would be natural to select test cases in such a way that the most critical system parts are tested most thoroughly.

It is important to realize that the weight of a failure cannot be extracted from a purely behavioral model, as it may depend in an essential way on the particular application of the implementation under test (IUT). The importance of the same bug may vary considerably between, say, its occurrence as part of an electronic game, and that as part of the control of a nuclear power plant.

Overview. This paper introduces a semantic approach for test coverage that aims to overcome the two points mentioned above. Our point of departure is a WFM that assigns a weight to each potential error in an implementation. We define our coverage measures relative to these WFMs. Since WFMs are augmented specifications, our coverage framework qualifies as black box.

Since WFMs are infinite semantic objects, we need to represent them finitely if we want to model them or use them in algorithms. We provide such representations by fault automata (Section 4). Fault automata are rooted in ioco test

theory [15] (recapitulated in Section 3), but their principles apply to a much wider setting.

We provide two ways of deriving WFMs from fault automata, namely the finite depth WFMs (Section 4.1) and the discounted WFMs (Section 4.2). The coverage measures obtained for these fault automata are invariant under behavioral equivalence. For both fault models, we provide algorithms that calculate and optimize test coverage (Section 5). These can all be studied as optimization problems in a linear algebraic setting. In particular, we compute the (total, absolute and relative) coverage of a test suite w.r.t. a weighted fault model (WFM).

We apply our theory to a small chat protocol (Section 6) and end by providing conclusions and suggestions for further research (Section 7). Due to space restrictions, we refer the reader to [5] for the full version of this paper.

2 Coverage measures in weighted fault models

Preliminaries. Let L be any set. Then L^* denotes the set of all finite sequences over L, which we also call *traces* over L. The empty sequence is denoted by ε and $|\sigma|$ denotes the length of a trace $\sigma \in L^*$. We use $L^+ = L^* \setminus \{\varepsilon\}$. For σ, $\rho \in L^*$, we say that σ is a *prefix* of ρ and write $\sigma \sqsubseteq \rho$, if $\rho = \sigma\sigma'$ for some $\sigma' \in L^*$. If σ is a prefix of ρ, then ρ is a *suffix* of σ. We call σ a *proper prefix* of ρ and ρ a *proper suffix* of σ if $\sigma \sqsubseteq \rho$, but $\sigma \neq \rho$.

We denote by $\mathcal{P}(L)$ the power set of L and for any function $f : L \to \mathbb{R}$, we use the convention that $\sum_{x\in\emptyset} f(x) = 0$ and $\prod_{x\in\emptyset} f(x) = 1$.

2.1 Weighted fault models

A weighted fault model specifies the desired behavior of a system by not only providing the correct system traces, but by also giving the severity of the erroneous traces. In this section, we work with a fixed action alphabet L.

Definition 1. *A* weighted fault model (WFM) *over L is a function $f : L^* \to \mathbb{R}^{\geq 0}$ such that $0 < \sum_{\sigma\in L^*} f(\sigma) < \infty$.*

Thus, a WFM f assigns a non-negative error weight to each trace $\sigma \in L^*$. If $f(\sigma) = 0$, then σ represents correct system behavior; if $f(\sigma) > 0$, then σ represents incorrect behavior and $f(\sigma)$ denotes the severity of the error. So, the higher $f(\sigma)$, the worse the error. We sometimes refer to traces $\sigma \in L^*$ with $f(\sigma) > 0$ as *error traces* and traces with $f(\sigma) = 0$ as *correct traces* in f.

We require the total error weight $\sum_{\sigma\in L^*} f(\sigma)$ to be finite and non-zero, in order to define coverage measures relative to the total error weight.

2.2 Coverage measures

This section abstracts from the exact shape of test cases and test suites. Given a WFM f over action alphabet L, we only use that a test is a trace set, $t \subseteq L^*$; and a test suite is a collection of trace sets, $T \subseteq \mathcal{P}(L^*)$. In this way we define the absolute and relative coverage w.r.t. f of a test and for a test suite. Moreover, our coverage measures apply in all settings where test cases can be characterized as

trace sets (in which case test suites can be characterized as collections of trace sets). This is a.o. true for tests in TTCN [7], ioco test theory [15] and FSM testing [17, 9].

Definition 2. *Let $f : L^* \to \mathbb{R}^{\geq 0}$ be a WFM over L, let $t \subseteq L^*$ be a trace set and let $T \subseteq \mathcal{P}(L^*)$ be a collection of trace sets. We define*

- $abscov(t, f) = \sum_{\sigma \in t} f(\sigma)$ *and* $abscov(T, f) = abscov(\cup_{t \in T} t, f)$
- $totcov(f) = abscov(L^*, f)$
- $relcov(t, f) = \frac{abscov(t,f)}{totcov(f)}$ *and* $relcov(T, f) = \frac{abscov(T,f)}{totcov(f)}$

The coverage of a test suite T w.r.t. f measures the total weight of the errors that can be detected by tests in T. The absolute coverage $abscov(T, f)$ simply accumulates the weights of all error traces in T. Note that the weight of each trace is counted only once, since one test case is enough to detect the presence of an error trace in an IUT. The relative coverage $relcov(T, f)$ yields the error weight in T as a fraction of the weight of all traces in L^*. Since absolute (coverage) numbers have meaning only if they are put in perspective of a maximum or average; we advocate that the relative coverage yields a good indication for the quality of a test suite.

Completeness of a test suite can easily be expressed in terms of coverage.

Definition 3. *A test suite $T \subseteq \mathcal{P}(L^*)$ is* complete *w.r.t. a WFM $f : L^* \to \mathbb{R}^{\geq 0}$ if $relcov(T, f) = 1$.*

3 Test cases in labeled input-output transition systems

This section recalls some basic theory about test derivation from labeled input-output transition systems, following ioco testing theory [15]. It prepares for the next section that treats an automaton-based formalism for specifying WFMs.

3.1 Labeled input-output transition systems

Definition 4. *A* labeled input-output transition system (LTS) *$\mathcal{A}$ is a tuple $\langle S, s^0, L, \Delta \rangle$, where*

- *S is a finite set of states*
- *$s^0 \in S$ is the initial state*
- *L is a finite action alphabet. We assume that $L = L^I \cup L^O$ is partitioned (i.e. $L^I \cap L^O = \emptyset$) into a set L^I of* input labels *(also called input actions or inputs) and a set L^O of* output labels *L^O (also called output actions or outputs). We denote elements of L^I by $a?$ and elements of L^O by $a!$*
- *$\Delta \subseteq S \times L \times S$ is the transition relation. We require Δ to be deterministic, i.e. if $(s, a, s'), (s, a, s'') \in \Delta$, then $s' = s''$. The input transition relation Δ^I is the restriction of Δ to $S \times L^I \times S$ and the output transition relation Δ^O is the restriction of Δ to $S \times L^O \times S$. We write $\Delta(s) = \{(a, s') \mid (s, a, s') \in \Delta\}$ and similarly for $\Delta^I(s)$ and $\Delta^O(s)$. We denote by $outdeg(s) = |\Delta(s)|$ the outdegree of state s, i.e. the number of transitions leaving s*

We denote the components of $\mathcal{A}$ by $S_\mathcal{A}$, $s^0_\mathcal{A}$, $L_\mathcal{A}$, and $\Delta_\mathcal{A}$. We omit the subscript $\mathcal{A}$ if it is clear from the context.

We have required $\mathcal{A}$ to be deterministic only for technical simplicity. This is not a real restriction, since we can always determinize $\mathcal{A}$. We can also incorporate quiescence (i.e. the absence of outputs), by adding a self loop $s \xrightarrow{\delta} s$ labeled with a special label δ to each quiescent state s, i.e. each s with $\Delta^O(s) = \emptyset$ and considering δ as an output action. But, since quiescence is not preserved under determinization, we must first determinize and then add quiescence.

Example 1. Figure 1 a) presents a LTS of a MP3 player: if the user pushes the play-button, a song should be played. In b), we see the extension with quiescence. Since δ is not enabled in state q_1, we explicitly forbid the absence of outputs in q_1, i.e. a song must be played. The double circles represent the initial state.

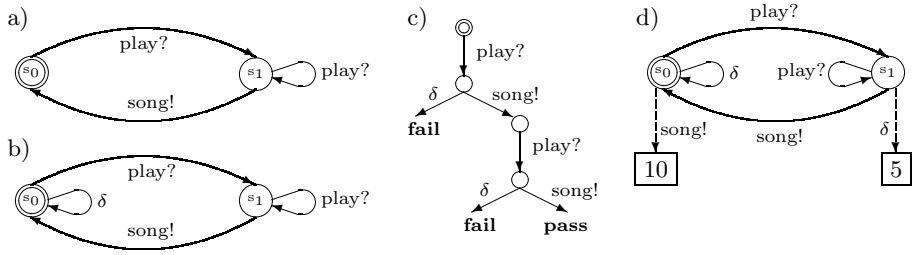

Fig. 1. A MP3 player: 1) its specification as a LTS; b) its extension with quiescence; c) a possible tests case t_1; and d) its fault automaton

We introduce the usual language theoretic concepts for LTSs.

Definition 5. *Let $\mathcal{A}$ be a LTS, then*

- *A* path *in $\mathcal{A}$ is a finite sequence $\pi = s_0 a_1 s_1 \dots s_n$ such that $s_0 = s^0$ and, for all $1 \leq i \leq n$, we have $(s_{i-1}, a_i, s_i) \in \Delta$. We denote by $paths_\mathcal{A}$ the set of all paths in $\mathcal{A}$ and by $last(\pi) = s_n$ the last state of π.*
- *The* trace *of a path π, $trace(\pi)$, is the sequence $a_1 a_2 \dots a_n$ of actions occurring in π. We write $traces_\mathcal{A} = \{trace(\pi) | \pi \in paths_\mathcal{A}\}$ for the set of all traces in $\mathcal{A}$.*
- *Let $\sigma \in L^*$ be any trace, not necessarily one from $\mathcal{A}$. We write $reach^k_\mathcal{A}(\sigma)$ for the set of states that can be reached in $\mathcal{A}$ in exactly k steps by following σ, i.e. $s' \in reach^k_\mathcal{A}(s)$ if $|\sigma| = k$ and there is a path $\pi \in paths_\mathcal{A}$ such that $trace(\pi) = \sigma$ and $last(\pi) = s'$. We write $reach_\mathcal{A}(\sigma)$ for the set of states that can be reached via trace σ in any number of steps, i.e. $reach_\mathcal{A}(\sigma) = \cup_{k \in \mathbb{N}} reach^k_\mathcal{A}(\sigma)$; we write $reach^k_\mathcal{A}$ for the set of states that can be reached in k number of steps, by following any trace, i.e. $reach^k_\mathcal{A} = \cup_{\sigma \in L^*} reach^k_\mathcal{A}(\sigma)$; and $reach_\mathcal{A} = \cup_{\sigma \in L^*} reach_\mathcal{A}(\sigma)$ for the set of all reachable states in $\mathcal{A}$.*

As before, we leave out the subscript $\mathcal{A}$ if it is clear from the context.

Definition 6. *Let $\mathcal{A}$ be a LTS and $s \in S$ be a state in $\mathcal{A}$, then $\mathcal{A}[s]$ denotes the LTS $\langle S, s, L, \Delta \rangle$.*

Thus, $\mathcal{A}[s]$ is the same as $\mathcal{A}$, but with s as initial state. This notation allows us to speak of paths, traces, etc, in $\mathcal{A}$ starting from a state that is not the initial state. For instance, $paths_{\mathcal{A}[s]}$ denotes the set of paths starting from state s.

3.2 Test cases

Test cases for LTSs are based on ioco test theory [15]. As in TTCN, ioco test cases are adaptive. That is, the next action to be performed (observe the IUT, stimulate the IUT or stop the test) may depend on the test history, that is, the trace observed so far. If, after a trace σ, the tester decides to stimulate the IUT with an input $a?$, then the new test history becomes $\sigma a?$; in case of an observation, the test accounts for all possible continuations $\sigma b!$ with $b! \in L^O$ an output action. Ioco theory requires that tests are "fail fast", i.e. stop after the discovery of the first failure, and never fail immediately after an input. If $\sigma \in traces_{\mathcal{A}}$, but $\sigma a? \notin trace_{\mathcal{A}}$, then the behavior after $\sigma a?$ is not specified in s, leaving room for implementation freedom. Formally, a test case consists of the set of all possible test histories obtained in this way.

Definition 7.
- *A* test case *(or test) t for a LTS $\mathcal{A}$ is a finite, prefix-closed subset of $L^*_{\mathcal{A}}$ such that*
  - *if $\sigma a? \in t$, then $\sigma b \notin t$ for any $b \in L$ with $a? \neq b$*
  - *if $\sigma a! \in t$, then $\sigma b! \in t$ for all $b! \in L^O$*
  - *if $\sigma \notin traces_{\mathcal{A}}$, then no proper suffix of σ is contained in t*

  We denote the set of all tests for $\mathcal{A}$ by $\mathcal{T}(\mathcal{A})$.
- *The* length *$|t|$ of test t is the length of the longest trace in t, i.e. $|t| = \max_{\sigma \in t} |\sigma|$. We denote by $\mathcal{T}^k(\mathcal{A})$ the set of all tests for $\mathcal{A}$ with length k.*

Example 2. Figure 1 c) shows a test case for the MP3 player from Figure 1, represented as a tree and augmented with verdicts pass and fail. The prefix closed trace set is obtained by taking all traces in the tree.

Since each test of $\mathcal{A}$ is a set of traces, we can apply Definition 2 and speak of (absolute, total and relative) coverage of a test case (or a test suite) of $\mathcal{A}$, w.r.t to a WFM f. However, not all WFMs are consistent with the interpretation that traces of $\mathcal{A}$ represent correct system behavior, and that tests are "fail fast" and do not fail after an input.

Definition 8. *Let $\mathcal{A}$ be a LTS and let $f : L^* \to \mathbb{R}^{\geq 0}$ be a WFM. Then f is* consistent with *$\mathcal{A}$ if $L = L_{\mathcal{A}}$ and for all $\sigma \in L^*_{\mathcal{A}}$ we have*

- *If $\sigma \in traces_{\mathcal{A}}$, then $f(\sigma) = 0$ (correct traces have weight 0).*
- *$f(\sigma a?) = 0$ (no failure occurs after an input).*
- *If $f(\sigma) > 0$ then $f(\sigma\rho) = 0$ for all $\rho \in L^+_{\mathcal{A}}$ (at most one failure per trace).*

The following result states that the set containing all possible test cases has complete coverage.

Theorem 1. *Let $\mathcal{A}$ be a LTS and f be a WFM consistent with $\mathcal{A}$. Then, the set $\mathcal{T}(\mathcal{A})$ of all test cases for $\mathcal{A}$ is complete w.r.t. f.*

4 Fault automata

Weighted fault models are infinite, semantic objects. This section introduces *fault automata*, which provide a syntactic format for specifying WFMs. A fault

automaton is a LTS $\mathcal{A}$ augmented with a state weight function r. The LTS $\mathcal{A}$ is the behavioral specification of the system, i.e. its traces represent the correct system behaviors. Hence, these traces will be assigned error weight 0; traces not in $\mathcal{A}$ are erroneous and get an error weight through r, as explained below.

Definition 9. *A* fault automaton (FA) $\mathcal{F}$ *is a pair* $\langle \mathcal{A}, r \rangle$*, where* $\mathcal{A}$ *is a LTS and* $r : S \times L^O \to \mathbb{R}^{\geq 0}$*. We require that, if* $r(s, a!) > 0$*, then there is no* $a!$*-successor of* s *in* $\mathcal{F}$*, i.e. there is no* $s' \in S$ *such that* $(s, a!, s') \in \Delta$*. We define* $\overline{r} : S \to \mathbb{R}^{\geq 0}$ *as* $\overline{r}(s) = \sum_{a \in L^O(s)} r(s, a)$*. Thus,* $\bar{r}$ *accumulates the weight of all the erroneous outputs in a state. We denote the components of* $\mathcal{F}$ *by* $\mathcal{A}_\mathcal{F}$ *and* $r_\mathcal{F}$ *and leave out the subscripts* $\mathcal{F}$ *if it is clear from the context. We lift all concepts and notations (e.g. traces, paths, etcetera) that have been defined for LTSs to FAs.*

Example 3. Figure 1 d) presents a FA for our MP3 example. We give error weight 5 if in state q_0 a song is played; and weight 10 if in state q_1 no song occurs.

We wish to construct a WFM f from the FA $\mathcal{F}$, using r to assign weights to traces not in $\mathcal{A}$. If there is no outgoing $a!$-transition in s, then the idea is that, for a trace σ ending in s, the (incorrect) trace $\sigma a!$ gets weight $r(s, a!)$. Doing so, however, could cause the total error weight $totcov(f)$ to be infinite.

We consider two solutions to this problem. First, *finite depth WFMs* (Section 4.1) consider, for a given $k \in \mathbb{N}$, only faults in traces of length k or smaller. Second, *discounted WFMs* (Section 4.2) obtain finite total coverage through discounting, while considering error weight in all traces. The solution presented here are only two potential solutions, there are many other ways to derive a WFM from a fault automaton.

4.1 Finite depth weighted fault models

As said before, the finite depth model derives a WFM from a FA $\mathcal{F}$, for a given $k \in \mathbb{N}$, by ignoring all traces of length longer than k, i.e. by putting their error weight to 0. For all other traces, the weight is obtained via the function r. If σ is a trace of $\mathcal{F}$ ending in s, but $\sigma a!$ is not a trace in $\mathcal{F}$, then $\sigma a!$ gets weight $r(s, a!)$.

Definition 10. *Given a FA* $\mathcal{F}$*, and a number* $k \in \mathbb{N}$*, we define the function* $f^k_\mathcal{F} : L^* \to \mathbb{R}^{\geq 0}$ *by*

$$f^k_\mathcal{F}(\varepsilon) = 0 \qquad f^k_\mathcal{F}(\sigma a) = \begin{cases} r(s, a) & \text{if } s \in reach^k_\mathcal{F}(\sigma) \wedge a \in L^O \\ 0 & \text{otherwise} \end{cases}$$

Note that this function is uniquely defined because $\mathcal{F}$ is deterministic, so that there is at most one s with $s \in reach^k_\mathcal{F}(\sigma)$. Also, if $f^k_\mathcal{F}(\sigma a) = r(s, a) > 0$, then $\sigma \in traces_\mathcal{F}$, but $\sigma a \notin traces_\mathcal{F}$.

The following proposition states that $f^k_\mathcal{F}$ is a WFM consistent with $\mathcal{F}$, provided that $\mathcal{F}$ contains as most one state with a positive accumulated weight and that is reachable within k steps.

Proposition 1. *Let* $\mathcal{F}$ *be a FA, and* $k \in \mathbb{N}$*. If there is an* $i \leq k$ *and a state* $s \in reach^i_\mathcal{F}$ *with* $\overline{r}(s) > 0$*, then* $f^k_\mathcal{F}$ *is a WFM consistent with* $\mathcal{F}$*.*

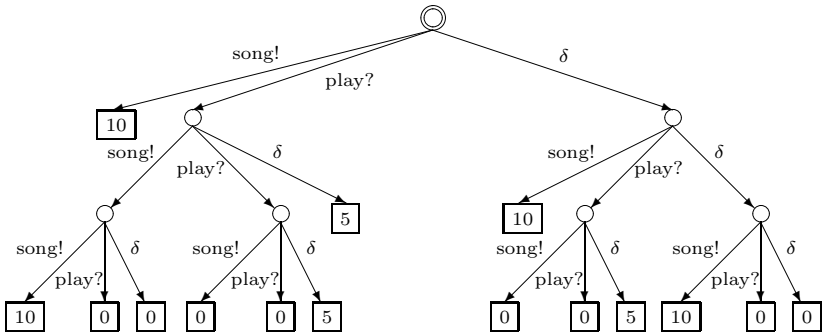

Fig. 2. Function $f^k_{\mathcal{F}}$, for $k = 3$ and $\mathcal{F}$ from Figure 1 b)

Example 4. Given the FA $\mathcal{F}$ from Figure 1 d), Figure 2 shows the function $f^k_{\mathcal{F}}$ for $k = 3$. Using t the test presented in Figure 1 c), we can obtain $abscov(t, f^k_{\mathcal{F}}) = 5$.

4.2 Discounted weighted fault models

While finite depth WFMs achieve finite total coverage by considering finitely many traces, discounted WFMs take into account the error weights of all traces. To do so, only finitely many traces may have weight greater than ϵ, for any $\epsilon > 0$. One way to do this is by discounting: lowering the weight of a trace proportional to its length. The rationale behind this is that errors in the near future are worse than errors in the far future, and hence, the latter should have higher weights.

In its basic form, a discounted WFM f for an FA $\mathcal{F}$ sets the weight of a trace $\sigma a!$ to $\alpha^{|\sigma|} r(s, a!)$, for some discount factor $\alpha \in (0, 1)$. If we take α small enough, then one can easily show that $\sum_{\sigma \in L^*} f(\sigma) < \infty$. To be precise, we take $\alpha < \frac{1}{d}$, where d is the branching degree of $\mathcal{F}$ (i.e. $d = \max_{s \in S} outdeg(s)$). Indeed, let $\alpha d < 1$ and $M = \max_s r(s, a)/\alpha$. Then $f(\sigma) \leq \alpha^{|\sigma|} M$. Since there are at most d^k traces of length k in $\mathcal{F}$, it follows that

$$\sum_{\sigma \in L^*} f(\sigma) = \sum_{k \in \mathbb{N}} \sum_{\sigma \in L^k} \alpha^k M \leq \sum_{k \in \mathbb{N}} d^k \alpha^k M = \frac{M}{1 - d\alpha} < \infty$$

To obtain more flexibility, we allow the discount to vary per transition. That is, we work with a discount function $\alpha : S \times L \times S \to \mathbb{R}^{\geq 0}$ that assigns a positive weight to each transition of $\mathcal{F}$. Then we discount the trace $a_1 \dots a_k$ obtained from the path $s_0 a_1 s_1 \dots s_k$ by $\alpha(s_0, a_1, s_1)\alpha(s_1, a_2, s_2) \cdots \alpha(s_{k-1}, a_k, s_k)$. The requirement that α is small enough now becomes: $\sum_{a \in L, s' \in S} \alpha(s, a, s') < 1$, We can even be more flexible and, in the sum above, do not range over states in which all paths are finite, as in these states we have finite total coverage anyway. Thus, if $Inf_{\mathcal{F}}$ is the set of all states in $\mathcal{F}$ with at least one outgoing infinite path, we require for all states s: $\sum_{a \in L, s' \in Inf_{\mathcal{F}}} \alpha(s, a, s') < 1$.

Definition 11. *Let $\mathcal{F}$ be a FA. The set $Inf_{\mathcal{F}} \subseteq S_{\mathcal{F}}$ of states with at least one infinite path is defined as $Inf_{\mathcal{F}} = \{s \in S \mid \exists \pi \in paths_{\mathcal{F}[s]} \,.\, |\pi| > |S|\}$.*

Definition 12. *Let $\mathcal{F}$ be a FA. Then a* discount function *for $\mathcal{F}$ is a function $\alpha : S_{\mathcal{F}} \times L_{\mathcal{F}} \times S_{\mathcal{F}} \to \mathbb{R}^{\geq 0}$ such that*

- *For all $s, s' \in S_{\mathcal{F}}$, and $a \in L_{\mathcal{F}}$ we have $\alpha(s, a, s') = 0$ iff $(s, a, s') \notin \Delta_{\mathcal{F}}$.*

- *For all $s \in S_{\mathcal{F}}$, we have:* $\sum_{a \in L_{\mathcal{F}}, s' \in Inf_{\mathcal{F}}} \alpha(s, a, s') < 1$.

Definition 13. *Let α be a discount function for the FA $\mathcal{F}$. Given a path $\pi = s_0 a_1 \dots s_n$ in $\mathcal{F}$, we define $\alpha(\pi)$ as $\prod_{i=1}^{n} \alpha(s_{i-1}, a_i, s_i)$.*

Definition 14. *Let $\mathcal{F}$ be a FA, $s \in S$, and α a discount function for $\mathcal{F}$. We define the function $f^{\alpha}_{\mathcal{F}} : L^* \to \mathbb{R}^{\geq 0}$ by*

$$f^{\alpha}_{\mathcal{F}}(\varepsilon) = 0$$

$$f^{\alpha}_{\mathcal{F}}(\sigma a) = \begin{cases} \alpha(\pi) \cdot r(s, a) & \text{if } s \in reach_{\mathcal{F}}(\sigma) \wedge a \in L^O \wedge trace(\pi) = \sigma \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathcal{F}$ is deterministic, there is at most one π with $trace(\pi) = \sigma$ and at most one $s \in reach(\sigma)$. Hence, the function above is uniquely defined.

The following proposition states that $f^{\alpha}_{\mathcal{F}}$ is a WFM consistent with $\mathcal{F}$, if $\mathcal{F}$ contains as most one reachable state with a positive accumulated weight.

Proposition 2. *Let $\mathcal{F}$ be a FA and α a discount function for $\mathcal{F}$. If there is a state $s \in reach_{\mathcal{F}}$ with $\overline{r}(s) > 0$, then $f^{\alpha}_{\mathcal{F}}$ is a WFM consistent with $\mathcal{F}$.*

Example 5. Figure 3 presents function $f^{\alpha}_{\mathcal{F}}$ for $\mathcal{F}$ from Figure 1 b) and $\alpha(s, a, s') = \gamma$ for every transition $(s, a, s') \in \Delta$. Using t the test presented in Figure 1 c), we can obtain $abscov(t, f^{\alpha}_{\mathcal{F}}) = \gamma^2 5$.

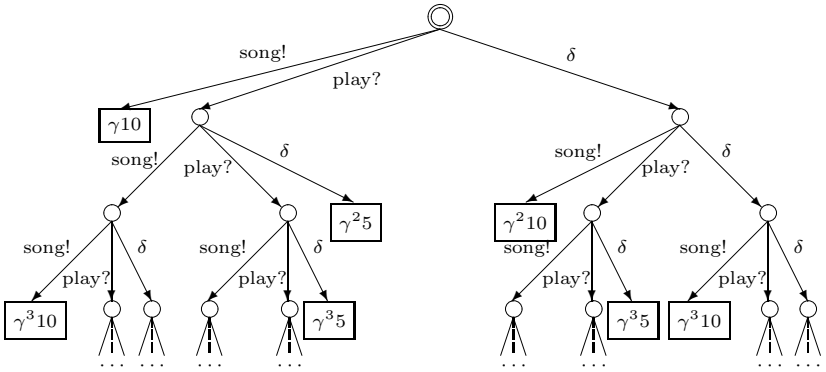

Fig. 3. Function $f^{\alpha}_{\mathcal{F}}$ for $\mathcal{F}$ from Figure 1 b) and $\alpha(s, a, s') = \gamma$

It is not difficult to see that our coverage notions are truly semantic in that they are invariant under $\overline{r}$-preserving bisimilarity. More precisely, suppose states s, s' in $\mathcal{F}$ are bisimilar, and that bisimilar states are required to have the same $\overline{r}$-value (i.e. $\overline{r}(s) = \overline{r}(s')$). Then $f^{k}_{\mathcal{F}[s]} = f^{k}_{\mathcal{F}[s']}$ and $f^{\alpha}_{\mathcal{F}[s]} = f^{\alpha}_{\mathcal{F}[s']}$ for all α and k. We refer the reader to [5] for more details.

4.3 Calibration

Discounting weighs errors in short traces more than in long traces. Thus, if we discount too much, we may obtain very high test coverage just with a few short test cases. The calibration result (Theorem 2) presented in this section shows

that, for any FA $\mathcal{F}$ and any $u > 0$, we can choose the discounting function in such a way that test cases of a given length k or longer are needed to achieve test coverage higher than a coverage bound $1 - u$. That is, we show that for any given k and u, there exists a discount function α such that the relative coverage of all test cases of length k or shorter is less than u. This means that, to get coverage higher than $1 - u$, one needs test cases longer than k.

For technical reasons, the weight assignment function of a FA have to be fair, i.e. all states in *Inf* must be able to reach some state with a positive weight.

Definition 15. *A FA $\mathcal{F}$ has* fair weight assignment *if for all $s \in Inf_{\mathcal{F}}$, there exists state $s' \in reach_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$.*

Theorem 2. *Let $\mathcal{F}$ be a FA with fair weight assignment. Then there exists a family of discount functions $\{\alpha_u\}_{u\in(0,1)}$ for $\mathcal{F}$ such that for all $k \in \mathbb{N}$ we have $\lim_{u\downarrow 0} relcov(\mathcal{T}^k(f^{\alpha_u}_{\mathcal{F}}), f^{\alpha_u}_{\mathcal{F}}) = 0$.*

5 Algorithms

This section represents various algorithms for computing and optimizing coverage for a given FA, interpreted under the finite depth or discounted weighted fault model.

In particular, Section 5.1 presents algorithms to calculate the absolute coverage in a test suite of a given FA. In Section 5.2 we give algorithms that yield the total coverage in a weighted fault model derived from a FA. Section 5.3 provides two optimization algorithms. The first one finds a test case of length k with maximal coverage; the second one finds a test suite with n test cases of length k and maximal coverage.

We use the following notation. Recall that $\mathcal{F}[s]$ denotes the FA that is the same as $\mathcal{F}$, but with s as initial state. When $\mathcal{F}$ is clear from the context, we write respectively f^k_s and f^α_s for the weighted fault models $f^k_{\mathcal{F}[s]}$ and $f^\alpha_{\mathcal{F}[s]}$ derived from $\mathcal{F}$. Moreover, given a FA $\mathcal{F} = \langle \mathcal{A}, r\rangle$, we write $A_{\mathcal{F}}$ for the multi-adjacency matrix of $\mathcal{A}$, containing at position (s, s') the number of edges between s and s', i.e. $(A_{\mathcal{F}})_{ss'} = \sum_{a:(s,a,s')\in\Delta} 1$. If α is a discount function for $\mathcal{F}$, then $A^\alpha_{\mathcal{F}}$ is a weighted version of $A_{\mathcal{F}}$, i.e. $(A^\alpha_{\mathcal{F}})_{ss'} = \sum_{a\in L} \alpha(s, a, s')$. We omit the subscript $\mathcal{F}$ if it is clear from the context.

5.1 Absolute coverage in a test suite

Given test suite T, a FA $\mathcal{F}$, and either a discounting function α for $\mathcal{F}$ or a number k, we desire to compute $abscov(T, f) = abscov(\cup_{t\in T} t, f)$, where $f = f^k_{\mathcal{F}}$ or $f^\alpha_{\mathcal{F}}$. Given two tests t and t' and an action a, we write at for $\{a\sigma \mid \sigma \in t\}$ and $t + t'$ for the union $t \cup t'$. We call a supertest (Stest) the union of any number of tests.

Now, we can write each test as $t = \varepsilon$; or $t = at_1$ in case a is an input; or $t = b_1t_1 + \cdots + b_nt_n$ when $b_1, \cdots, b_n$ are all output actions of $\mathcal{F}$. Each supertest can be written as $a_1t'_1 + \cdots + a_kt'_k + b_1t''_1 + \cdots + b_nt''_n$ where a_i are inputs and b_i are all outputs and t'_i, t''_j are supertests.

To compute the union $\cup_{t\in T} t$, we recursively merge all tests in T into a supertest using the infix operator $\uplus : Stest\times \text{ test} \to Stest$. Then we add the error weights of all traces in $\cup_{t\in T} t$ via the function ac.

Merging of tests. Let t' be a Stest, $t' = a_1 t'_1 + \cdots + a_k t'_k + b_1 t''_1 + \cdots + b_n t''_n$ and t be a test, then $t = \varepsilon$ or $t = at_1$ or $t = b_1 t'_1 + \cdots + b_n t'_n$

$t' \uplus t =$

$$\begin{cases} a_1 t'_1 + \cdots + a_j(t'_j \uplus t_1) + \cdots + a_k t'_k + b_1 t''_1 + \cdots + b_n t''_n & \text{if } t = at_1 \wedge a = a_j \\ a_1 t'_1 + \cdots + a_k t'_k + b_1(t''_1 \uplus t_1) + \cdots + b_n(t''_n \uplus t_n) & \text{if } t = b_1 t'_1 + \cdots + b_n t'_n \\ t' + t & \text{otherwise} \end{cases}$$

We write $\uplus\{t_1, t_2, \ldots, t_n\}$ for $t_1 \uplus t_2 \uplus \cdots \uplus t_n$.

Absolute coverage in a Stest. Given a Stest t of $\mathcal{F}$ and a state s on $\mathcal{F}$, then

$$ac(\varepsilon, s) = 0 \qquad ac(t, s) = \textstyle\sum_{i=1}^{n} aux(a_i t_i, s)$$

$$\text{where } aux(a_i t_i, s) = \begin{cases} \alpha(s, a_i, \delta(s, a_i)) \cdot ac(t_i, \delta(s, a_i)) & \text{if } a_i \in \delta(s) \\ r(a_i, s) & \text{otherwise} \end{cases}$$

The correctness of this algorithm is stated in the following theorem.

Theorem 3. *Given a FA $\mathcal{F}$, a state $s \in V$, a number $k \in \mathbb{N}$, a function $\alpha : S \times L \times S \to [0,1]$ and T a test suite, then*

- *If α is a discount function for $\mathcal{F}$, then $abscov(T, f_s^\alpha) = ac(\uplus T, s)$*
- *If $k \geq \max_{t \in T} |t|$ and $\alpha(s, a, s') = 1$ for all transitions (s, a, s') in $\mathcal{F}$, then $abscov(T, f_s^k) = ac(\uplus T, s)$.*

5.2 Total coverage

Total coverage in discounted FA. Given a FA $\mathcal{F}$, a state $s \in S$ and a discounting function α for $\mathcal{F}$, we desire to calculate $totcov(f_s^\alpha) = \sum_{\sigma \in L^*} f_s^\alpha(\sigma)$. We assume that from each state in $\mathcal{F}$ we can reach at least one error state (i.e. $\forall s \in S : \exists s' \in reach_{\mathcal{F}[s]} : \overline{r}(s) > 0$). In this way, f_s^α is a WFM for every s.

The basic idea behind the computation method is that the function $tc : S \to [0,1]$ ("total coverage") given by $s \mapsto totcov(f_s^\alpha)$ satisfies the following set of equations.

$$tc(s) = \overline{r}(s) + \sum_{a \in L, s' \in S} \alpha(s, a, s') \cdot tc(s') = \overline{r}(s) + \sum_{s' \in S} A_s^\alpha s' \cdot tc(s') \qquad (*)$$

These equations express that the total coverage in state s equals the weight $\overline{r}(s)$ of all immediate errors in s, plus the weights in all successors s' in s, discounted by $\sum_{a \in L} \alpha(s, a, s')$. Using matrix-vector notation, we obtain $tc = \overline{r} + A^\alpha \cdot tc$. In [5] it is shown that the matrix $I - A^\alpha$ is invertible. Thus, we obtain the following result; in particular, tc is the unique solution of the equations (*) above.

Theorem 4. *Let $\mathcal{F}$ be a FA such that for all $s \in S$ there exists a state $s' \in reach_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$, and let α be a discount function for $\mathcal{F}$. Then $tc = (I - A^\alpha)^{-1} \cdot \overline{r}$.*

Complexity. The complexity of the method above is dominated by matrix inversion, which can be computed in $O(|S|^3)$ with Gaussian elimination, $O(|S|^{log_2 7})$ with Strassen's method or even faster with more sophisticated techniques.

Total coverage in finite depth FA. Given a FA $\mathcal{F}$, a state $s \in S$ and a depth $k \in \mathbb{N}$, we desire to compute $totcov(f_s^k) = \sum_{\sigma \in L^*} f_s^k(\sigma)$. We assume that from each state, there is at least one error reachable in k steps (i.e. $\forall s \in S : \exists s' \in reach^k_{\mathcal{F}[s]} : \overline{r}(s') > 0$). This makes that f_s^k is a weighted fault model for any s.

The basic idea behind the computation method is that the function $tc_k : S \to [0,1]$ given by $s \mapsto totcov(f_s^k)$ satisfies the following recursive equations.

$$tc_0(s) = 0$$

$$tc_{k+1}(s) = \overline{r}(s) + \sum_{(a,s') \in \Delta(s)} tc_k(s') = \overline{r}(s) + \sum_{a \in L, s' \in S} A_{s,s'} \cdot tc_k(s')$$

Or, in matrix-vector notation, we have $tc_0 = 0$ and $tc_{k+1} = \overline{r} + A \cdot tc_k$.

Theorem 5. *Let $\mathcal{F}$ be a FA, a state $s \in S$ and $k \in \mathbb{N}$. If $\forall s \in S : \exists s' \in reach^k_{\mathcal{F}[s]} : \overline{r}(s') > 0$, then $tc_k = \sum_{i=0}^{k-1} A^i \cdot \overline{r}$.*

Complexity. Using Theorem 5 with sparse matrix multiplication, or iterating the equations just above it, tc_k can be computed in time $O(k \cdot |\Delta| + |S|)$.

Remark 1. A similar method to the one above can be used to compute the weight of all tests of length k in the discounted weighted fault model, i.e. $abscov(T^k, f_s^\alpha)$, for T^k (i.e. the set of all tests of length k in $\mathcal{F}$).

The recursive equations for computing $abscov(T^k, f_s^\alpha)$ are obtained by replacing A in Equation (*) by A_α. Since $I - A^\alpha$ is (unlike $I - A$) invertible, the analogon of Theorem 5 becomes $abscov(T^k, f_s^\alpha) = \sum_{i=0}^{k-1} (A_\alpha)^i \overline{r} = (I - A^\alpha)^{-1} \cdot (I - (A^\alpha)^k) \cdot \overline{r}$.

Relative coverage Combining the algorithms for computing total and absolute coverage from the previous sections, one easily computes $relcov(T, f) = \frac{abscov(T,f)}{totcov(f)}$ for a testsuite T and $f = f_s^k$ or $f = f_s^\alpha$.

5.3 Optimization

Optimal coverage in a test case. Given a FA $\mathcal{F}$ and a length k, we compute the best test case with length k (i.e. the one with highest coverage). We treat the finite depth and discounted model at once by fixing, in the finite depth model $\alpha(s, a, s') = 1$ if (s, a, s') is a transition in Δ and $\alpha(s, a, s') = 0$ otherwise. A function *alpha* is call *extended discount function* if it is a discount function or it is obtained from a finite depth model in the presented previous way.

The optimization method is again based on recursive equations. We write $acopt_k(s) = \max_{t \in \mathcal{T}^k} \{abscov(t, s)\}$ ("optimal absolute coverage"). Consider a test case of length $k+1$ that in state s applies an input $a?$ and in the successor state s' applies the optimal test of length k. The (absolute) coverage of this test case is $\alpha(s, a?, s') \cdot acopt_k(s')$. The best coverage that we can obtain by stimulating the IUT is given by $\max_{(a?,s') \in \Delta^I(s)} \alpha(s, a?, s') \cdot acopt_k(s')$.

Now, consider the test case of length $k+1$ that in state s observes the IUT and in each successor state s' applies the optimal test of length k. The coverage of this test case is $\overline{r}(s) + \sum_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \cdot acopt_k(s')$. The optimal test $acopt(s)$ of length $k+1$ is obtained from $acopt_k$ by selecting from these options (i.e. inputing an action $a?$ or observing) the one with the highest coverage.

Theorem 6. *Let $\mathcal{F}$ be a FA, α be an extended discount function, and $k \in \mathbb{N}$ test length. Then $acopt_k$ satisfies the following recursive equations. $acopt_0(s) = 0$ and $acopt_{k+1}(s) = \max\{\overline{r}(s) + \sum_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \cdot acopt_k(s'), \max_{(a?,s') \in \Delta^I(s)} \alpha(s, a?, s') \cdot acopt_k(s')\}$.*

Complexity. Based on Theorem 6, we can compute $acopt_k$ in time $O(k(|S|+|\Delta|))$.

Shortest test case with high coverage. We can use the above method not only to compute the test case of a fixed length k with optimal coverage, but also to derive the shortest test case with coverage higher than a given bound c. We iterate the equations in Theorem 6 and stop as soon as we achieve coverage higher than c, i.e. at the first n with $acopt_k(s) > c$.

We have to take care that the bound c is not too high, i.e. higher than what is achievable with a single test case. In the finite depth model, this is easy: if the test length is the same as c then we can stop, since this is the longest test we can have. In the discounted model, however, we have to ensure that c is strictly smaller than the supremum of the coverage of all tests in single test case.

Let $mw(s) = supp_{t \in \mathcal{T}} abscov(t, s)$, i.e. the maximal absolute weight of a single test case. Then mw is again characterized by a set of equations.

Theorem 7. *Let $\mathcal{F}$ be a FA, and α be a discount function for $\mathcal{F}$. Then mw is the unique solution of the following set of equations: $mw(s) = \max\{\max_{(a?,s') \in \Delta^I(s)} \alpha(s, a?, s') \cdot mw(s'), \overline{r}(s) + \sum_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \cdot mw(s')\}$.*

The solution of these equations can be found by linear programming (LP).

Theorem 8. *Let $\mathcal{F}$ be a FA, and α be a discount function. Then mw is the optimal solution of the following LP problem: minimize $\sum_{s \in S} mw(s)$ subject to*

$$mw(s) \geq \alpha(s, a?, s') \cdot mw(s') \qquad (a?, s') \in \Delta^I(s)$$

$$mw(s) \geq r(s) + \sum_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \cdot mw(s') \qquad s \in S$$

Complexity. The above LP problem contains $|S|$ variables and $|S|+|\Delta^I|$ inequalities. Thus, solving this problem is polynomial in $|S|$, $|S| + |\Delta^I|$ and the length of the binary encoding of the coefficients [14]. In practice, the exponential time simplex method outperforms existing polynomial time algorithms.

Optimal coverage in n cases. The first algorithm in this section for computing the best test case of length k can be extended to a method for computing the best n test cases with optimal coverage: the previous algorithm picks the best test case with length k. To pick the second best test case, we apply the same procedure, except that we exclude the first choice from all possible options, for the third best choice, we exclude the previous two, etc. See [5] for more details.

6 Application: a chat protocol

This section applies our theory to a practical example, namely a chat protocol, also used as a conference protocol [2]. This protocol provides a multi-cast service

	tc	tck, $k=2$	rck, $k=2$	tck, $k=4$	rck, $k=4$	tck, $k=50$	rck, $k=50$
α_1	99.134	89.750	91%	97.171	98%	99.134	100%
α_2	511.369	130.607	25%	239.025	47%	510.768	100%
α_3	743.432	132.652	18%	249.320	34%	733.540	99%

Fig. 4. Total coverage (tc); absolute (tck) and relative coverage (rck) of the test suite containing all tests of length k

	test t_1^k	test t_2^k	test t_3^k	test t_4^k	test t_5^k	test t_6^k	test t_7^k	test t_8^k	test t_9^k	test t_{10}^k	suite T^k
$k=30$	15.3%	4.6%	14.0%	5.3%	15.3%	4.6%	14.2%	8.5%	15.3%	4.9%	63.1%
$k=35$	14.1%	15.3%	15.3%	8.5%	8.6%	5.3%	15.3%	8.5%	8.5%	4.9%	69.1%
$k=40$	5.3%	14.0%	14.2%	15.3%	5.3%	14.1%	15.3%	5.3%	14.0%	15.3%	72.8%
$k=45$	5.0%	8.5%	14.0%	5.0%	8.5%	15.3%	4.9%	15.3%	4.5%	14.2%	47.2%
$k=50$	5.3%	14.2%	5.3%	4.9%	14.0%	5.3%	14.2%	5.3%	14.0%	15.3%	54.2%

Fig. 5. Relative coverage, as a percentage, of tests generated by TorX, using α_2

to users engaged in a chat session. Each user can send messages to and receive messages from all other partners participating in the same chat session. The chat participants are dynamic, as the chat service allows them to join and leave the chat at any moment in time. Different chats can exist at the same time, but each user can only participate in at most one chat at a time.

Based on the LTS model in [2], we have created an FA $\mathcal{F}$ for this protocol. Our model considers two chat sessions and three users and has 39 states and 95 transitions. The complete model and the transition weight function can be found in [5]. We interpret $\mathcal{F}$ as a discounted WSM under different discount functions, α_1, α_2 and α_3. If $\theta = (s, a, s')$ is a transition in $\mathcal{F}$ leaving from state s with outdegree d, we use $\alpha_1(\theta) = \frac{1}{8}$; $\alpha_2(\theta) = \frac{1}{d} - \frac{1}{100}$; and $\alpha_3(\theta) = \frac{1}{d} - \frac{1}{10000}$.

Figure 4 gives the total coverage in $\mathcal{F}$ (column 1) and the absolute (columns 2, 4, 6) and relative (columns 3, 5, 7) coverage of the test suites containing all tests of length k, for $k = 2, 4, 50$ and $\alpha_1, \alpha_2, \alpha_3$. These results were obtained by applying the first (total coverage in discounted FA) and third algorithm (Remark 1) from Section 5.2. We used Maple 9.5 to resolve the matrix equations in these algorithms.

Figure 5 displays the relative coverage for test suites that have been generated automatically with TorX, using discount function α_2. For test lengths $k = 30, 35, 40, 45, 50$, TorX has generated a test suite T^k, consisting of 10 tests $t_1^k, \ldots t_{10}^k$ of length k. We used Algorithm 5.1 to calculate the relative coverage of T^k. Figure 5 lists the coverage of each individual test t_i^k as well as for the test suites T^k.

The running times of all computations were very small, in the order of a few seconds. Note how the figures show the influence of the discount factor and the test length on the coverage numbers.

7 Conclusions and future research

Semantic notions of test coverage have long been overdue, while they are much needed in the selection, generation and optimization of test suites. In this paper, we presented semantic coverage notions based on WFMs. We introduced fault

automata, FA, to syntactically represent (a subset of) WFMs and provided algorithms to compute and optimize test coverage. This approach is purely semantic since replacing a FA with a semantically equivalent one (i.e. $\overline{r}$-preserving bisimilar) leaves the coverage unchanged. Our experiments with the chat protocol indicate that our approach is feasible for small protocols. Larger case studies should evaluate the applicability of this framework for more complex systems.

Our weighted fault models are based on (adaptive) ioco test theory. We expect to be easy to adapt our approach to different settings, such as FSM testing or on-the-fly testing. Furthermore, our optimization techniques use test length as an optimality criterion. To accommodate more complex resource constraints (e.g time, costs, risks/probability) occurring in practice, it is relevant to extend our techniques with these attributes. Since these fit naturally within our model and optimization problems subject to costs, time and probability are well-studied, we expect that such extensions are both feasible and useful.

References

1. Ball, T. A theory of predicate-complete test coverage and generation. In *Proceedings of FMCO'04* (2004), pp. 1–22.
2. Belinfante, A., Feenstra, J., Vries, R., Tretmans, J., Goga, N., Feijs, L., Mauw, S., and Heerink, L. Formal test automation: A simple experiment. In *Int. Workshop on Testing of Communicating Systems 12* (1999), pp. 179–196.
3. Belinfante, A., Frantzen, L., and Schallhart, C. Tools for test case generation. In *Model-Based Testing of Reactive Systems* (2004), pp. 391–438.
4. Brandán Briones, L., and Brinksma, E. A test generation framework for *quiescent* real-time systems. In *FATES'04* (2004), pp. 64–78.
5. Brandán Briones, L., Brinksma, E., and Stoelinga, M. A semantic framework for test coverage (extended version). Tech. Rep. TR-CTIT-06-24, Centre for Telematics and Information Technology, University of Twente, 2006.
6. Campbell, C., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N., and Veanes, M. Model-based testing of object-oriented reactive systems. Tech. Rep. MSR-TR-2005-59, 2005.
7. ETSI. Es 201 873-6 v1.1.1 (2003-02). Methods for testing and specification (mts). In *The Testing and Test Control Notation version 3: TTCN-3 Control Interface (TCI). ETSI Standard* (2003).
8. Jard, C., and Jéron, T. TGV: theory, principles and algorithms. *STTT 7*, 4 (2005), 297–315.
9. Lee, D., and Yannakakis, M. Principles and methods of testing finite state machines - A survey. In *Proceedings of the IEEE* (1996), vol. 84, pp. 1090–1126.
10. Myers, G. *The Art of Software Testing.* Wiley & Sons, 1979.
11. Myers, G., Sandler, C., Badgett, T., and Thomas, T. *The Art of Software Testing.* Wiley & Sons, 2004.
12. Nachmanson, L., Veanes, M., Schulte, W., Tillmann, N., and Grieskamp, W. Optimal strategies for testing nondeterministic systems. In *International Symposium on Software Testing and Analysis* (2004), ACM Press, pp. 55–64.
13. Nicola, R., and Hennessy, M. Testing equivalences for processes. In *Proceedings ICALP* (1983), vol. 154.
14. Tardos, E. A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5*, 3 (1985), 247–255.

15. Tretmans, J. Test generation with inputs, outputs and repetitive quiescence. *Software-Concepts and Tools 17*, 3 (1996), 103–120.
16. Tretmans, J., and Brinksma, E. TorX: Automated model-based testing. In *First European Conference on Model-Driven Software Engineering* (2003).
17. Ural, H. Formal methods for test sequence generation. *Computer Communications Journal 15*, 5 (1992), 311–325.